

EEEEEEEEEEEEEEEE	MMM	MMM	UUU	UUU	LLL	AAAAAAAA	TTTTTTTTTTTTTT
EEEEEEEEEEEEEEEE	MMM	MMM	UUU	UUU	LLL	AAAAAAAA	TTTTTTTTTTTTTT
EEEEEEEEEEEEEEEE	MMM	MMM	UUU	UUU	LLL	AAAAAAAA	TTTTTTTTTTTTTT
EEE	MMMMMM	MMMMMM	UUU	UUU	LLL	AAA	TTT
EEE	MMMMMM	MMMMMM	UUU	UUU	LLL	AAA	TTT
EEE	MMMMMM	MMMMMM	UUU	UUU	LLL	AAA	TTT
EEE	MMM	MMM	UUU	UUU	LLL	AAA	TTT
EEE	MMM	MMM	UUU	UUU	LLL	AAA	TTT
EEE	MMM	MMM	UUU	UUU	LLL	AAA	TTT
EEEEEEEEEEEEEEEE	MMM	MMM	UUU	UUU	LLL	AAA	TTT
EEEEEEEEEEEEEEEE	MMM	MMM	UUU	UUU	LLL	AAA	TTT
EEEEEEEEEEEEEEEE	MMM	MMM	UUU	UUU	LLL	AAA	TTT
EEE	MMM	MMM	UUU	UUU	LLL	AAAAAAAAAAAAAAAA	TTT
EEE	MMM	MMM	UUU	UUU	LLL	AAAAAAAAAAAAAAAA	TTT
EEE	MMM	MMM	UUU	UUU	LLL	AAAAAAAAAAAAAAAA	TTT
EEE	MMM	MMM	UUU	UUU	LLL	AAA	TTT
EEE	MMM	MMM	UUU	UUU	LLL	AAA	TTT
EEE	MMM	MMM	UUU	UUU	LLL	AAA	TTT
EEEEEEEEEEEEEEEE	MMM	MMM	UUUUUUUUUUUUUU	UUUUUUUUUUUUUU	LLLLLLLLLLLLLLLL	AAA	TTT
EEEEEEEEEEEEEEEE	MMM	MMM	UUUUUUUUUUUUUU	UUUUUUUUUUUUUU	LLLLLLLLLLLLLLLL	AAA	TTT
EEEEEEEEEEEEEEEE	MMM	MMM	UUUUUUUUUUUUUU	UUUUUUUUUUUUUU	LLLLLLLLLLLLLLLL	AAA	TTT

Sym

---

CMP

DEC

DEC

DEC

DEC

EXE

EXE

EXE

MMG

MOV

PRB

SYS

VAX

VAX

VAX

VAX

VAX

VAX

VAX

VAX

VAX

VAX

VAX

VAX

VAX

VAX

VAX

VAX

VAX

VAX

VAX

VAX

VAX

VAX

VAX

VAX

VAX

VAX

```

LL          IIIIII          SSSSSSSS
LL          IIIIII          SSSSSSSS
LL          II             SS
LL          II             SS
LL          II             SS
LL          II             SS
LL          II             SSSSSS
LL          II             SSSSSS
LL          II             SS
LL          II             SS
LL          II             SS
LL          II             SS
LLLLLLLLLLLL IIIIIII SSSSSSSS
LLLLLLLLLLLL IIIIIII SSSSSSSS

```



VAX\$HANDLER  
Table of contents

C 15  
- Condition Handlers for VAX-11 Instruct 16-SEP-1984 01:36:04 VAX/VMS Macro V04-00

Page 0

(2)	101	Declarations
(3)	185	VAX\$AL_DELTA_PC_TABLE
(4)	267	VAX\$ACVIOLAT - Modify Access Violation
(5)	418	Reserved Operand or Addressing Mode Exception
(6)	503	VAX\$REFLECT_FAULT - Reflect Fault to User
(10)	809	VAX\$REFLECT_TRAP - Reflect Arithmetic Traps
(11)	944	VAX\$REFLECT_TO_VMS - Let VMS Reflect the Exception

VAX  
V04



```
0000 1 .TITLE VAX$HANDLER - Condition Handlers for VAX-11 Instruction Emulator
0000 2 .IDENT /V04-000/
0000 3
0000 4
0000 5 *****
0000 6 *
0000 7 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 8 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 9 * ALL RIGHTS RESERVED.
0000 10 *
0000 11 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 12 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 13 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 14 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 15 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 16 * TRANSFERRED.
0000 17 *
0000 18 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 19 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 20 * CORPORATION.
0000 21 *
0000 22 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 23 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 24 *
0000 25 *
0000 26 *****
0000 27
0000 28
0000 29 ++
0000 30 Facility:
0000 31
0000 32 VAX-11 Instruction Emulator
0000 33
0000 34 Abstract:
0000 35
0000 36 This module contains all interfaces between the VAX-11 instruction
0000 37 emulator and the VAX/VMS condition handling facility. That is, this is
0000 38 the only module in the entire VAX-11 instruction emulator package that
0000 39 has any knowledge of VMS-specific aspects of condition handling. All
0000 40 exception knowledge contained in other modules is defined by the
0000 41 VAX-11 Architecture (microVAX subset). If this emulator is to be used
0000 42 in an environment other than VMS, this is the only module that needs
0000 43 to be changed.
0000 44
0000 45 Note that control flows through this module in two directions. On the
0000 46 one hand, certain exceptions such as access violation can occur while
0000 47 the emulator is executing. These exceptions are reported by VMS to a
0000 48 special access violation routine in this module. These routines and
0000 49 instruction-specific exception handling routines manipulate this
0000 50 exception so that it appears to have occurred at the site of the
0000 51 reserved instruction, rather than within the emulator.
0000 52
0000 53 Other exceptions such as reserved operand abort for illegal decimal
0000 54 string length or decimal overflow trap are detected by emulator
0000 55 routines. These must be reported by the emulator to VMS, again in such
0000 56 a way that it appears to the running code that the exception occurred
0000 57 at the site of the reserved instruction.
```



```
0000 58 :
0000 59 : Environment:
0000 60 :
0000 61 :     These routines run at any access mode, at any IPL, and are AST
0000 62 :     reentrant.
0000 63 :
0000 64 : Author:
0000 65 :
0000 66 :     Kathleen D. Morse
0000 67 :
0000 68 : Creation Date
0000 69 :
0000 70 :     17 August 1982
0000 71 :
0000 72 : Modified by:
0000 73 :
0000 74 :     V01-005 LJK0036      Lawrence J. Kenah      17-Jul-1984
0000 75 :     Fix INSV bug in REFLECT_FAULT that was causing PSL bits
0000 76 :     to be incorrectly cleared.
0000 77 :
0000 78 :     V01-004 LJK0022      Lawrence J. Kenah      9-Feb-1984
0000 79 :     Final cleanup pass. Eliminate stack switch logic. VMS now
0000 80 :     transfers control here in the mode of the exception and not
0000 81 :     in kernel mode. Add logic to distinguish a stack that
0000 82 :     contains a signal array from a stack that also contains a
0000 83 :     mechanism array and a CHF argument list.
0000 84 :
0000 85 :     V01-003 LJK0019      Lawrence J. Kenah      25-Jan-1984
0000 86 :     Revise the way that exceptions are handled. Eliminate code
0000 87 :     that is not being used.
0000 88 :
0000 89 :     V01-002 LJK0002      Lawrence J. Kenah      15-Mar-1983
0000 90 :     Handle software detected exceptions from decimal and EDITPC
0000 91 :     emulation. Modify initial and final dispatching to use registers
0000 92 :     instead of stack space for parameter passing. Include stack
0000 93 :     format generated by microVAX exceptions.
0000 94 :
0000 95 :     V01-001 Original      Kathleen D. Morse      17-Aug-1982
0000 96 :     Intercept access violations and machine checks and conditionally
0000 97 :     dispatch them to emulator for modification. Reflect exceptions
0000 98 :     from emulator to regular VMS exception dispatcher.
0000 99 :--
```



```
0000 101          .SUBTITLE      Declarations
0000 102
0000 103 ; Include files:
0000 104
0000 105          $CHFDEF          ; Offsets into signal array
0000 106          $OPDEF           ; Symbolic names for opcodes
0000 107          $PRDEF           ; Get definitions of processor registers
0000 108          $PSLDEF          ; Define bit fields in PSL
0000 109          $SRMDEF          ; Arithmetic trap codes
0000 110          $SSDEF           ; Status codes in VMS
0000 111
0000 112          .NOCROSS          ; No cross reference for these
0000 113          .ENABLE          SUPPRESSION ; No symbol table entries either
0000 114
0000 115          STACK_DEF        ; Stack storage of exception parameters
0000 116          PACK_DEF         ; Stack usage for reflecting exceptions
0000 117
0000 118          .DISABLE          SUPPRESSION ; Turn on symbol table again
0000 119          .CROSS            ; Cross reference is OK now
0000 120
0000 121 ; Macro definitions
0000 122
0000 123          .MACRO DELTA_PC TABLE ENTRY      OPCODE
0000 124          SIGN_EXTEND OP$ 'OPCODE', ...OPCODE
0000 125          .IIF LESS THAN <...OPCODE - OPCODE_BASE>,-
0000 126          .ERROR          ; Opcode not supported by emulator
0000 127          .IIF GREATER <...OPCODE - OPCODE_MAX>,-
0000 128          .ERROR          ; Opcode not supported by emulator
0000 129          .NOCROSS
0000 130          .ENABLE          SUPPRESSION
0000 131          OPCODE' _DEF      ; Register usage for OPCODE instruction
0000 132          .DISABLE          SUPPRESSION
0000 133          .CROSS
0000 134          . = DELTA_PC TABLE_BASE + <...OPCODE-OPCODE_BASE>
0000 135          .BYTE OPCODE' B DELTA_PC
0000 136          .ENDM DELTA_PC_TABLE_ENTRY
0000 137
0000 138          .MACRO CMP_L_TO A      SRC,DSTADR,DST,TYPE=B
0000 139          .IF BLANK DST
0000 140          PUSH A'TYPE          DSTADR
0000 141          CMPL SRC,(SP)+
0000 142          .IF_FALSE
0000 143          MOVA'TYPE          DSTADR,DST
0000 144          CMPL SRC,DST
0000 145          .ENDC
0000 146          .ENDM
0000 147
0000 148 ; Symbol definitions
0000 149
0000 150          JSB_ABSOLUTE = <^X9Fa8> ! OP$_JSB
0000 151          JSB_SIZE = 6 ; Size of JSB @#VAX$xxxxxx instruction
0000 152
0000 153 ; External declarations
0000 154
0000 155          .DISABLE          GLOBAL
0000 156
0000 157          .EXTERNAL VAX$EXIT_EMULATOR ; Return PC to dispatcher
```



```
0000 158
0000 159      .EXTERNAL      EXE$REFLECT      ; Entry point in VMS
0000 160
0000 161      ::: THE FOLLOWING DECLARATIONS ARE NOT NEEDED UNLESS WE DEFINE ENTRY POINTS
0000 162      ::: INTO THE EMULATOR THROUGH THE SYSTEM SERVICE VECTOR PAGES.
0000 163      :::
0000 164      .EXTERNAL      SYSS$VAX_BEGIN,- ; Lower and upper bounds of JSB
0000 165      :::          SYSS$VAX_END      ; entry points in vector page
0000 166
0000 167 ; Default Addressing Mode
0000 168
0000 169      .DEFAULT          DISPLACEMENT , WORD
0000 170
0000 171 ; PSECT Declarations:
0000 172
0000 173 ;      This label defines the beginning of the emulator image
0000 174
00000000 175      .PSECT _VAX$$$BEGIN PIC, USR, CON, REL, LCL, SHR, EXE, RD, NOWRT, PAGE
0000 176
0000 177 VAX$BEGIN:
0000 178
0000 179 ;      This label locates the end of the emulator image
0000 180
00000000 181      .PSECT _VAX$_END  PIC, USR, CON, REL, LCL, SHR, EXE, RD, NOWRT, BYTE
0000 182
0000 183 VAX$END:
```



[illegible]



```
0001 237 DELTA_PC_TABLE_ENTRY CMPP3
003E 238 DELTA_PC_TABLE_ENTRY CMPP4
0040 239 DELTA_PC_TABLE_ENTRY CVTLP
0002 240 DELTA_PC_TABLE_ENTRY CVTPL
003F 241 DELTA_PC_TABLE_ENTRY CVTPS
0011 242 DELTA_PC_TABLE_ENTRY CVTPT
002D 243 DELTA_PC_TABLE_ENTRY CVTSP
0012 244 DELTA_PC_TABLE_ENTRY CVTTP
002F 245 DELTA_PC_TABLE_ENTRY DIVP
0030 246 DELTA_PC_TABLE_ENTRY MOVP
003D 247 DELTA_PC_TABLE_ENTRY MULP
002E 248 DELTA_PC_TABLE_ENTRY SUBP4
002B 249 DELTA_PC_TABLE_ENTRY SUBP6
002C 250
002C 251 ; Don't forget good old EDITPC
002C 252
002C 253 DELTA_PC_TABLE_ENTRY EDITPC
0041 254
0041 255 ; Locate the table through entry 0
0041 256
00000008 0041 257 VAX$AL_DELTA_PC_TABLE == DELTA_PC_TABLE_BASE + <0 - OPCODE_BASE>
0041 258
0041 259 ; Finally, set the location counter to the end of the table
0041 260
00000044 0041 261 . = DELTA_PC_TABLE_BASE + DELTA_PC_TABLE_SIZE
0044 262
0044 263 ; The code that does useful work is put into this section.
0044 264
00000000 265 .PSECT _VAX$CODE PIC, USR, CON, REL, LCL, SHR, EXE, RD, NOWRT, QUAD
```

```
0000 267 .SUBTITLE VAX$ACVIOLAT - Modify Access Violation
0000 268
0000 269 :+ Functional Description:
0000 270
0000 271 This routine receives control from the exception dispatcher in VMS
0000 272 after the mechanism and signal arrays have been moved from the kernel
0000 273 stack to the stack of the mode in which the access violation occurred.
0000 274 This routine determines whether the exception occurred inside the
0000 275 emulator and, if so, passes control to an instruction-specific routine
0000 276 that performs further consistenct checks.
0000 277
0000 278 The purpose of all this work is to allow exceptions that occur inside
0000 279 the emulator to be modified before being passed on to the user. There
0000 280 are two important pieces of this modification.
0000 281
0000 282 The exception PC is changed from a PC within the emulator to
0000 283 the PC of the instruction that caused the emulator to be
0000 284 invoked in the first place.
0000 285
0000 286 Any stack usage by the emulator is dissolved. This allows
0000 287 the exception stack to be placed directly on top of the
0000 288 user's stack usage, removing any overt indication that an
0000 289 emulator is being used.
0000 290
0000 291 Although the only exception that can be modified in this fashion is an
0000 292 access violation, the routine is written in terms of a general signal
0000 293 array to allow expansion at a future time.
0000 294
0000 295 Input Parameters:
0000 296
0000 297 If this exception is one that the emulator is capable of modifying,
0000 298 then R10 must contain the address of an instruction-specific routine
0000 299 to store context information in the general registers before passing
0000 300 control back to this module.
0000 301
0000 302 Although VMS places the mechanism array directly underneath the
0000 303 argument list and, with one intervening longword, places the signal
0000 304 array directly underneath that, these lists display general signal and
0000 305 mechanism arrays.
0000 306
0000 307 00(SP) - Return PC in VMS exception dispatcher
0000 308 04(SP) - Argument count (always 2)
0000 309 08(SP) - Address of signal array
0000 310 12(SP) - Address of mechanism array
0000 311
0000 312 The mechanism array looks like this
0000 313
0000 314 00(mech) - Argument count (always 4)
0000 315 04(mech) - Value of FP when exception occurred
0000 316 08(mech) - Depth value (initially -3)
0000 317 12(mech) - Value of R0 when exception occurred
0000 318 16(mech) - Value of R1 when exception occurred
0000 319
0000 320 In the general case of an exception with M optional parameters, the
0000 321 signal array looks like this
0000 322
0000 323 00(signal) - Argument count (M+3)
```



```
0000 324 : 04(signal) - Exception name
0000 325 : 08(signal) - First optional parameter if M GTRU 0
0000 326 : .
0000 327 :
0000 328 : 4*M + 4(signal) - Last parameter if M GTRU 0
0000 329 : 4*M + 8(signal) - PC of faulting instruction
0000 330 : 4*M + 12(signal) - PSL at time of exception
0000 331 :
0000 332 : For the case of an access violation, which has two optional
0000 333 : parameters, the signal array takes this form.
0000 334 :
0000 335 : 00(signal) - Argument count (always 5)
0000 336 : 04(signal) - $$$_ACCVIO
0000 337 : 08(signal) - Access violation reason mask
0000 338 : 12(signal) - Inaccessible virtual address
0000 339 : 16(signal) - PC of faulting instruction
0000 340 : 20(signal) - PSL at time of exception
0000 341 :
0000 342 : The longword immediately following the exception PSL was on top of the
0000 343 : stack when the exception occurred.
0000 344 :
0000 345 : Output Parameters:
0000 346 :
0000 347 : If the exception passes the small number of tests performed here,
0000 348 : control is passed to the routine whose address is stored in R10
0000 349 : with the following output parameters.
0000 350 :
0000 351 : R0 - Address of top of stack (value of SP) when exception occurred
0000 352 : R1 - Exception PC
0000 353 :
0000 354 : R0 to R3 are saved on the stack to allow this routine and the
0000 355 : instruction-specific routines to do useful work without juggling
0000 356 : registers.
0000 357 :
0000 358 : 00(SP) - Value of R0 when exception occurred
0000 359 : 04(SP) - Value of R1 when exception occurred
0000 360 : 08(SP) - Value of R2 when exception occurred
0000 361 : 12(SP) - Value of R3 when exception occurred
0000 362 : 16(SP) - Return PC in VMS exception dispatcher
0000 363 : 20(SP) - Argument count (always 2)
0000 364 : 24(SP) - Address of signal array
0000 365 : 28(SP) - Address of mechanism array
0000 366 :
0000 367 : If the tests performed here determine that the exception does not
0000 368 : fit the pattern for a candidate to be modified, control is passed
0000 369 : back to VMS with an RSB instruction.
0000 370 :
0000 371 : Notes:
0000 372 :
0000 373 : Some of the code here assumes that it knows what a VMS exception stack
0000 374 : looks like. If it finds the stack in a different shape, it does not
0000 375 : continue. The reason is that, if VMS changes these insignificant pieces
0000 376 : of the stack, it may also change things that this routine relies on that
0000 377 : are more difficult to detect.
0000 378 : -
0000 379 :
0000 380 VAX$MODIFY_EXCEPTION::
```

```
53 7E 52 7D 0000 381      MOVQ    R2,-(SP)      ; Start with two scratch registers
    OC AE DE 0003 382      MOVAL    12(SP),R3      ; R3 locates argument list
    63 02 D1 0007 383      CMPL     #2,(R3)      ; Is argument count 2?
    41 12 000A 384      BNEQ     20$      ; Quit if not
52 04 A3 D0 000C 385      MOVL     CHF$SIGARGLST(R3),R2 ; R2 locates the signal array
53 08 A3 D0 0010 386      MOVL     CHF$MCHARGLIST(R3),R3 ; R3 locates the mechanism array
    63 04 D1 0014 387      CMPL     #4,CHF$MCH_ARGS(R3) ; Is this argument count 4?
    34 12 0017 388      BNEQ     20$      ; Quit if not
    7E 0C A3 7D 0019 389      MOVQ    CHF$MCH_SAVR0(R3),-(SP) ; Save original R0 and R1
    50 62 D0 001D 390      MOVL     CHF$SIG_ARGS(R2),R0 ; R0 contains signal argument count
51 FC A240 D0 0020 391      MOVL     <8-12>(R2)[R0],R1 ; Get exception PC
    0025 392
    0025 393 ; Check that exception PC and R10 are both within the bounds of the emulator.
    0025 394
53 0000'CF 9E 0025 395      MOVAB    VAX$BEGIN,R3      ; Put base address in convenient place
    53 51 D1 002A 396      CMPL     R1,R3      ; Is exception PC within this limit?
    1B 1F 002D 397      BLSSU     10$      ; Quit if PC is at smaller address
    53 5A D1 002F 398      CMPL     R10,R3      ; Is R10 within this limit?
    16 1F 0032 399      BLSSU     10$      ; Quit if R10 is at smaller address
    0034 400
    0034 401 ; Do the same checks with the ending address
    0034 402
53 0000'CF 9E 0034 403      MOVAB    VAX$END,R3      ; Put end address in convenient place
    53 51 D1 0039 404      CMPL     R1,R3      ; Is exception PC within this limit?
    OC 1E 003C 405      BGEQU     10$      ; Quit if PC is too large
    53 5A D1 003E 406      CMPL     R10,R3      ; Is R10 within this limit?
    07 1E 0041 407      BGEQU     10$      ; Quit if R10 is too large
    0043 408
    0043 409 ; Load R0 with the value of SP when the exception occurred
    0043 410
50 04 A240 DE 0043 411      MOVAL    <16-12>(R2)[R0],R0 ; Get top of stack at time of exception
    6A 17 0048 412      JMP      (R10)      ; Call instruction-specific routine
    004A 413
    50 8E 7D 004A 414 10$: MOVQ    (SP)+,R0      ; Restore R0 and R1
    52 8E 7D 004D 415 20$: MOVQ    (SP)+,R2      ; ... and R2 and R3
    05 0050 416      RSB
```



```
0051 418 .SUBTITLE Reserved Operand or Addressing Mode Exception
0051 419 :+
0051 420 : Functional Description:
0051 421 :
0051 422 : This routine receives control from the emulator routines for the
0051 423 : decimal instructions or EDITPC when those routines have detected a
0051 424 : condition that requires signalling of an exception. The exception can
0051 425 : either be a reserved addressing mode fault from CVTPL or a reserved
0051 426 : operand exception, either a fault or an abort, signalled from a
0051 427 : variety of places. This routine simply makes the stack look like the
0051 428 : stack on entry to VAX$REFLECT_FAULT and passes control to that routine
0051 429 : to perform the instruction backup in common code.
0051 430 :
0051 431 : Input Parameters:
0051 432 :
0051 433 : 00(SP) - Offset in packed register array to delta PC byte
0051 434 : 04(SP) - Return PC from VAX$xxxxxx routine
0051 435 :
0051 436 : Output Parameters:
0051 437 :
0051 438 : R0 - Locates return PC in middle of stack
0051 439 : R1 - Contains delta PC originally stored on top of stack
0051 440 :
0051 441 : 00(SP) - Saved R0
0051 442 : 04(SP) - Saved R1
0051 443 : 08(SP) - Saved R2
0051 444 : 12(SP) - Saved R3
0051 445 : 16(SP) - Size of signal array (always 3)
0051 446 : 20(SP) - Exception name (SS$_ROPRAND or SS$_RADRMOD)
0051 447 : 24(SP) - Place holder for PC of exception
0051 448 : 28(SP) - PSL of exception
0051 449 : 32(SP) - Offset to delta PC byte (no longer needed)
0051 450 : R0 -> 36(SP) - Return PC from VAX$xxxxxx routine
0051 451 :
0051 452 : Implicit Output:
0051 453 :
0051 454 : This routine exits by dropping into the VAX$REFLECT_FAULT routine
0051 455 : that decides the particular form the instruction backup will take.
0051 456 :
0051 457 : Notes:
0051 458 :
0051 459 : There are three ways that a reserved operand exception can occur.
0051 460 :
0051 461 : 1. Digit count of packed decimal string GTRU 31
0051 462 :
0051 463 : This is an abort where the PC points to the offending
0051 464 : decimal or EDITPC instruction.
0051 465 :
0051 466 : 2. Illegal numeric or sign digit detected by CVTSP or CVTTP
0051 467 :
0051 468 : An illegal numeric digit was detected by one of these
0051 469 : instructions or an illegal sign character was detected by
0051 470 : CVTSP. This exception is also not restartable.
0051 471 :
0051 472 : 3. Illegal EDITPC pattern operator
0051 473 :
0051 474 : The EDITPC decoder detected an illegal pattern operator. This
```

```
0051 475 : exception stores the intermediate state in registers and may
0051 476 : be restarted.
0051 477 :
0051 478 : A reserved addressing mode exception can only occur when the PC is
0051 479 : used as the destination operand in the CVTPL instruction.
0051 480 :-
0051 481 :
0051 482 .ENABLE LOCAL_BLOCK
0051 483
0051 484 VAX$RADRMOD::
0051 485 MOVPSL -(SP) ; Store exception PSL
0053 486 CLRL -(SP) ; Save space for the exception PC
0055 487 PUSHL #SS$_RADRMOD ; Store exception name
005B 488 BRB 10$ ; Join common code
005D 489
005D 490 VAX$ROPRAND::
005D 491 MOVPSL -(SP) ; Store exception PSL
005F 492 CLRL -(SP) ; Save space for the exception PC
0061 493 PUSHL #SS$_ROPRAND ; Store exception name
0067 494
0067 495 10$: PUSHL #3 ; Store signal array size
0069 496 PUSHR #^M<R0,R1,R2,R3> ; Store the usual registers
006B 497 MOVL 32(SP),R1 ; R1 gets delta PC offset
006F 498 MOVAL 36(SP),R0 ; R0 locates the return PC
0073 499 ; Drop through to VAX$REFLECT_FAULT
0073 500
0073 501 .DISABLE LOCAL_BLOCK
```

BCDEFGHIJKLMNOPQRSTUVWXYZBCDEFGHIJKLMNOPQRSTUVWXYZBCDEFGHIJKLMNOPQRSTUVWXYZ



```
0073 503 .SUBTITLE VAX$REFLECT_FAULT - Reflect Fault to User
0073 504
0073 505 :+ Functional Description:
0073 506
0073 507 : This routine reflects a fault (such as an access violation that
0073 508 : occurred inside the emulator) back to the user. The signal array, in
0073 509 : particular, the exception PC, is modified to point to the reserved
0073 510 : instruction or the JSB instruction into the emulator.
0073 511
0073 512 : Input Parameters:
0073 513
0073 514 : R0 - Address on stack of return PC
0073 515 : R1<7:0> - Byte offset from top of stack into saved register array
0073 516 : (R0..R3) where delta-PC will be stored if original path into
0073 517 : emulator was through a reserved instruction exception
0073 518 : R1<8> - distinguishes restartable exceptions (faults) from
0073 519 : exceptions that cannot be restarted (aborts)
0073 520 : R1<9> - distinguishes software generated exceptions from exceptions
0073 521 : detected by hardware, detoured through the emulator, and
0073 522 : modified by instruction-specific routines.
0073 523
0073 524 : Note that the condition codes in the exception PSL are significant for
0073 525 : faults, primarily to make the EDITPC illegal pattern operator
0073 526 : exception conform to the architecture.
0073 527
0073 528 : 00(SP) - Saved R0
0073 529 : 04(SP) - Saved R1
0073 530 : 08(SP) - Saved R2
0073 531 : 12(SP) - Saved R3
0073 532
0073 533 : 16(SP) - Number of additional longwords in signal array (called N)
0073 534 : 20(SP) - Exception name
0073 535
0073 536 : IF N GTRU 3 THEN
0073 537 : 24(SP) - First exception-specific parameter
0073 538 :
0073 539 :
0073 540 : <4*<N-2> + 16>(SP) - Last exception specific parameter
0073 541 :
0073 542 : <4*<N-1> + 16>(SP) - PC of exception
0073 543 : <4*N + 16>(SP) - PSL of exception
0073 544 :
0073 545 : <4*N + 16 + 4>(SP) - Instruction specific storage (no longer needed)
0073 546 :
0073 547 :
0073 548 : -04(R0) - Last longword of instruction specific storage
0073 549 :
0073 550 : (R0) - Return PC from VAX$xxxxxx routine in emulator
0073 551 :
0073 552 : There are three possibilities for the return PC. The action of this
0073 553 : routine depends on this return PC value.
0073 554 :
0073 555 : Case 1. (R0) - VAX$EXIT_EMULATOR
0073 556 :
0073 557 : This is the usual case where the emulator was entered as a result of
0073 558 : an emulated instruction exception. The signal array from the second
0073 559 : exception is put on top of the original exception array, the rest of
```



```
0073 560 : the stack is evaporated, and the exception is reflected to the user.
0073 561 :
0073 562 : The FPD bit in the exception PSL is set for those exceptions that are
0073 563 : restartable.
0073 564 :
0073 565 : Case 2. (R0) - Address of instruction following
0073 566 :
0073 567 : JSB @#VAX$xxxxxx
0073 568 :
0073 569 : In this case, the signal array that is passed back to the user is
0073 570 : reflected has the address of the JSB instruction as the exception PC.
0073 571 : The FPD bit is ALWAYS clear. Note that it is much more difficult (if
0073 572 : not impossible) to back up an arbitrary instruction that transfers
0073 573 : control to the emulator.
0073 574 :
0073 575 : Case 3. (R0) - Anything else
0073 576 :
0073 577 : In this case, the emulator was entered in some other way. Because
0073 578 : instruction state has already been modified, it is no longer possible to
0073 579 : simply reflect the secondary exception (as we did with unrecognized
0073 580 : exceptions in routine VAX$MODIFY_EXCEPTION). We add an additional longword
0073 581 : to the signal array (VAX$_ABORT) and report this slightly modified
0073 582 : exception to the original caller without modifying the return PC.
0073 583 :-
0073 584 :-
0073 585 VAX$REFLECT_FAULT::
52 80 D0 0073 586 MOVE (R0)+,R2 ; Get return PC from stack
0076 587 CMP_L_TO_A R2,VAX$EXIT_EMULATOR,R3
53 12 007E 588 BNEQU NO_SIGNAL_ARRAY ; Branch if no secondary signal array
0080 589
0080 590 ; If we drop through the branch, we are examining Case 1. We can use the signal
0080 591 ; array that already exists to hold the modified exception parameters.
```



```
0080 593 ; Case 1.      (R0) - VAX$EXIT_EMULATOR
0080 594 :
0080 595 :      More Input Parameters:
0080 596 :
0080 597 :      04(R0) - Opcode of reserved instruction
0080 598 :      08(R0) - PC of reserved instruction (old PC)
0080 599 :      12(R0) - First operand specifier (no longer needed)
0080 600 :      .
0080 601 :
0080 602 :      40(R0) - Eight operand specifier (place holder)
0080 603 :      44(R0) - PC of instruction following reserved instruction
0080 604 :      (new PC)
0080 605 :      48(R0) - PSL at time of exception
0080 606 :
0080 607 :      Output Parameters for Case 1:
0080 608 :
0080 609 :      R0 through R3 restored from top of stack
0080 610 :
0080 611 :      00(SP) - Size of signal array (called N)
0080 612 :      04(SP) - Exception name
0080 613 :
0080 614 :      IF N GTRU 3 THEN
0080 615 :          08(SP) - First exception-specific parameter
0080 616 :          .
0080 617 :          <4*<N-2>>(SP) - Last exception specific parameter
0080 618 :
0080 619 :          <4*<N-1>>(SP) - Old PC (PC of reserved instruction)
0080 620 :          <4*N>(SP) - PSL of second exception (FPD set)
0080 621 :
0080 622 :
0080 623 :
0080 624 :      .ENABLE      LOCAL_BLOCK
0080 625 :
0080 626 :      ; We need to capture the FPD information stored in R1 before that register
0080 627 :      ; is modified (or used as an index register).
0080 628 :
0080 629 :      BBCC #PACK_V_FPD,R1,10$ ; Branch if FPD bit remains clear
0080 630 :      BBSS #PSL$V_FPD,-
0080 631 :      EXCEPTION_PSL(R0),10$ ; Set FPD bit in exception PSL
0080 632 10$:  SUBL3 OLD_PC(R0),NEW_PC(R0),R2 ; Calculate delta PC
0080 633 :      MOVZBL R1,R3 ; Isolate delta-PC offset in R3
0080 634 :      MOVB R2,(SP)[R3] ; Store delta PC in one of R0..R3
0080 635 :      BBS #PACK_V_ACCVIO,R1,20$ ; Branch if more than signal array
0080 636 :
0080 637 :      ; In this case, the signal array is located immediately underneath the
0080 638 :      ; saved register array.
0080 639 :
0080 640 :      MOVZBL <PACK_L_SIGNAL_ARRAY+CHF$SIG_ARGS>(SP),R3
0080 641 :      ; Get signal array size
0080 642 :      MOVAL PACK_L_SIGNAL_ARRAY(SP)[R3],R2
0080 643 :      ; R2 points to exception PSL
0080 644 :      BRB 30$ ; Rejoin common code
0080 645 :
0080 646 :      ; In this case, there is other information on the stack between the saved
0080 647 :      ; register array and the signal array, namely a return address in VMS, an
0080 648 :      ; argument list that would have been passed to condition handlers had the
0080 649 :      ; exception not been detoured through this code, and a mechanism array.
```

05 51 08 E5 0080 629 BBCC #PACK\_V\_FPD,R1,10\$ ; Branch if FPD bit remains clear  
1B E2 0084 630 BBSS #PSL\$V\_FPD,-  
00 2C A0 0086 631 EXCEPTION\_PSL(R0),10\$ ; Set FPD bit in exception PSL  
52 28 A0 04 A0 C3 0089 632 10\$: SUBL3 OLD\_PC(R0),NEW\_PC(R0),R2 ; Calculate delta PC  
53 51 9A 008F 633 MOVZBL R1,R3 ; Isolate delta-PC offset in R3  
6E43 52 90 0092 634 MOVB R2,(SP)[R3] ; Store delta PC in one of R0..R3  
0B 51 09 E0 0096 635 BBS #PACK\_V\_ACCVIO,R1,20\$ ; Branch if more than signal array  
009A 636  
009A 637 ; In this case, the signal array is located immediately underneath the  
009A 638 ; saved register array.  
009A 639  
53 10 AE 9A 009A 640 MOVZBL <PACK\_L\_SIGNAL\_ARRAY+CHF\$SIG\_ARGS>(SP),R3  
009E 641 ; Get signal array size  
52 10 AE43 DE 009E 642 MOVAL PACK\_L\_SIGNAL\_ARRAY(SP)[R3],R2  
00A3 643 ; R2 points to exception PSL  
0B 11 00A3 644 BRB 30\$ ; Rejoin common code  
00A5 645  
00A5 646 ; In this case, there is other information on the stack between the saved  
00A5 647 ; register array and the signal array, namely a return address in VMS, an  
00A5 648 ; argument list that would have been passed to condition handlers had the  
00A5 649 ; exception not been detoured through this code, and a mechanism array.



```
00A5 650 ; All of this extra data is discarded and the exception as the modified
00A5 651 ; stack is passed back to the VMS exception dispatcher.
00A5 652
52 18 AE D0 00A5 653 20$: MOVL PACK_L_SIGNAL_ARRAY_POINTER(SP),R2
00A9 654 ; R2 locates signal array
52 53 62 D0 00A9 655 MOVL CHF$SIG_ARGS(R2),R3 ; Get signal array size
52 6243 DE 00AC 656 MOVAL CHF$SIG_ARGS(R2)[R3],R2
00B0 657 ; R2 points to exception PSL
51 2C A0 DE 00B0 658 30$: MOVAL EXCEPTION_PSL(R0),R1 ; R1 points to original PSL
00B4 659
00B4 660 ;+
00B4 661 ; R0 - Address of VAX$OPCDEC exception array
00B4 662 ; R1 - Address of PSL of original VAX$OPCDEC exception
00B4 663 ; R2 - Address of PSL in signal array of exception being backed up
00B4 664 ; R3 - Number of longwords in signal array
00B4 665 :-
00B4 666
FC A2 04 A0 D0 00B4 667 MOVL OLD_PC(R0),-4(R2) ; This extra MOVL actually saves code!
00B9 668
00B9 669 ; There are two more operations that need to be performed on the PSL that
00B9 670 ; will appear in the signal array. These operations are only significant for
00B9 671 ; faults. The affected PSL fields are defined to be UNPREDICTABLE in the case
00B9 672 ; of aborts. Rather than complicate the code with unnecessary branches,
00B9 673 ; however, the condition codes will be propagated and the TP bit will be
00B9 674 ; cleared in all cases.
00B9 675
61 04 00 62 F0 00B9 676 INSV (R2),#0,#4,(R1) ; Copy condition codes to PSL
00 61 1E E4 00BE 677 BBSC #PSL$V_TP,(R1),40$ ; Clear the TP bit
00C2 678
00C2 679 ; We now move a modified signal array down the stack, from the back (PSL) end
00C2 680 ; to the front (argument count) end. The PSL has already been moved before
00C2 681 ; the loop executes to allow the FPD bit to get set and to make the loop
00C2 682 ; count work correctly.
00C2 683
71 72 D0 00C2 684 40$: MOVL -(R2),-(R1) ; Move next longword
FA 53 F5 00C5 685 SOBGTR R3,40$ ; Check for any more
00C8 686
00C8 687 ; At the end of this loop, R1 points to what we want to report as the top of the
00C8 688 ; stack to the VMS exception dispatcher. We store R1 on the stack following the
00C8 689 ; saved R0 through R3 and set the new stack pointer as part of the POPR
00C8 690 ; instruction.
00C8 691
10 AE 51 D0 00C8 692 MOVL R1,PACK_L_SAVED_SP(SP) ; Load new SP underneath R0..R3 array
400F 8F BA 00CC 693 POPR #^M<R0,R1,R2,R3,SP> ; Restore registers and set SP
0079 31 00D0 694 BRW VAX$REFLECT_TO_VMS ; Use common exit path to VMS
```



```
00D3 696 : Case 2.      (R0) - Address of instruction following
00D3 697 :
00D3 698 :             JSB      @#VAX$xxxxxx
00D3 699 :
00D3 700 :             More Input Parameters:
00D3 701 :
00D3 702 :             There is nothing else of interest on the stack in this case.
00D3 703 :
00D3 704 :             Output Parameters for Case 2:
00D3 705 :
00D3 706 :             R0 through R3 restored from top of stack
00D3 707 :
00D3 708 :             00(SP) - Size of signal array (called N)
00D3 709 :             04(SP) - Exception name
00D3 710 :
00D3 711 :             IF N GTRU 3 THEN
00D3 712 :                 08(SP) - First exception-specific parameter
00D3 713 :
00D3 714 :
00D3 715 :                 <4*N-2>(SP) - Last exception specific parameter
00D3 716 :
00D3 717 :                 <4*N-1>(SP) - Address of JSB instruction
00D3 718 :                 <4*N>(SP) - PSL of second exception (FPD clear!)
00D3 719 :
00D3 720 :
00D3 721 : +
00D3 722 : This is either Case 2 or Case 3, depending on the instruction located by
00D3 723 : the return PC. If this is a return PC from a JSB @# into the emulator,
00D3 724 : then the instruction that we wish to examine lies six bytes before the
00D3 725 : location pointed to by R2.
00D3 726 : -
00D3 727 :
00D3 728 NO_SIGNAL_ARRAY:
00D3 729     CMPW      -6(R2),#JSB_ABSOLUTE      ; Is the opcode JSB @# ?
12F16 8F  FA A2  B1 00D9 730     BNEQ      UNKNOWN          ; Branch if not
16      12
00DB 731
00DB 732 ::: IF WE EVER INSTALL ENTRY POINTS INTO THE EMULATOR THROUGH THE SYSTEM
00DB 733 ::: SERVICE VECTOR PAGE, THE FOLLOWING CHECKS CAN BE TURNED ON. THE INTENT
00DB 734 ::: IS THAT, LIKE SYSTEM SERVICE CALLS, THE EMULATOR REFERENCES WOULD
00DB 735 ::: GENERATE JSB @# VAX$xxxxxx INSTRUCTIONS TO JMP INSTRUCTIONS IN THE
00DB 736 ::: VECTOR PAGES. IT IS ONLY THIS SET OF JSB INSTRUCTIONS THAT WOULD BE
00DB 737 ::: BACKED UP ACCORDING TO METHOD 2. ALL OTHER PATHS INTO THE EMULATOR
00DB 738 ::: WILL GENERATE EXCEPTIONS WITH AN EXCEPTION PC INSIDE THE EMULATOR ITSELF.
00DB 739
00DB 740 :::     MOVL      -4(R2),R2                ; Get destination of JSB @#
00DB 741 :::     CMP_L_TO_A  R2,SYSS$VAX_BEGIN,R3    ; Make lower bounds check
00DB 742 :::     BLSEQ     UNKNOWN          ; Branch if too small
00DB 743 :::     CMP_L_TO_A  R2,SYSS$VAX_END,R3    ; Make upper bounds check
00DB 744 :::     BGEQU     UNKNOWN          ; Branch if too large
00DB 745
00DB 746 : This is Case 2 as described above. It differs from Case 1 in two ways. There
00DB 747 : is no signal array on the stack underneath the stack that is overwritten.
00DB 748 : The FPD bit in the saved PSL is not set.
00DB 749
53      10 AE  9A 00DB 750     MOVZBL  <PACK_L_SIGNAL_ARRAY+CHF$L_SIG_ARGS>(SP),R3
53      53  D6 00DF 751
00DF 752     INCL      R3
```



VAX\$HANDLER  
V04-000

G 16

- Condition Handlers for VAX-11 Instruct 16-SEP-1984 01:36:04 VAX/VMS Macro V04-00 Page 17  
VAX\$REFLECT\_FAULT - Reflect Fault to Use 5-SEP-1984 00:45:37 [EMULAT.SRC]VAXHANDLR.MAR;1 (8)

F8	A2	52	1D	AE43	DE	00E1	753	MOVAL	PACK_L SIGNAL ARRAY(SP)[R3],R2 ; R2 points beyond exception PSL
		FC	A0	06	C3	00E6	754	SUBL3	#JSB_SIZE,-4(R0),-8(R2) ; PC of JSB becomes exception PC
			51	50	D0	00EC	755	MOVL	R0,RT ; Start writing signal array at return PC
				D1	11	00EF	756	BRB	40\$ ; Join common exit at top of loop



```
00F1 758 : Case 3.      (R0) - Anything else
00F1 759 :
00F1 760 :      This is a case where the emulator was entered in a nonstandard
00F1 761 :      way. This code has no way of creating an exception PC that
00F1 762 :      will cause the emulator to be reentered. Instead, the
00F1 763 :      exception is redefined to be VAX$_ABORT with the rest of the
00F1 764 :      original signal array comprising the exception parameters. The
00F1 765 :      PC is not modified so that knowledgeable code in the form of a
00F1 766 :      condition handler could conceivably restart such an exception
00F1 767 :      if it knew how to reenter the emulator.
00F1 768 :
00F1 769 :      More Input Parameters:
00F1 770 :
00F1 771 :      There is nothing else of interest on the stack in this case.
00F1 772 :
00F1 773 :      Output Parameters for Case 3:
00F1 774 :
00F1 775 :      R0 through R3 restored from top of stack
00F1 776 :
00F1 777 :      00(SP) - Size of new signal array (N + 1)
00F1 778 :      04(SP) - New exception name (VAX$_ABORT)
00F1 779 :      08(SP) - Original exception name
00F1 780 :
00F1 781 :      IF N GTRU 3 THEN
00F1 782 :          12(SP)      - First exception-specific parameter
00F1 783 :          .
00F1 784 :          <4*<N-2>+4>(SP) - Last exception specific parameter
00F1 785 :
00F1 786 :          <4*<N-1>+4>(SP) - "Anything else" (original return PC)
00F1 787 :          <4*N+4>(SP)   - PSL of second exception (FPD clear!)
00F1 788 :
00F1 789 :      :-
00F1 790 :
00F1 791 :      :+
00F1 792 :      : This is Case 3. The emulator was entered in some unorthodox fashion. We leave
00F1 793 :      : the exception PC alone but change the facility field in the exception name
00F1 794 :      : located in the signal array to VAX$_.
00F1 795 :      :-
00F1 796 :
00F1 797 :      UNKNOWN:
53 10 AE 9A 00F1 798      MOVZBL <PACK_L_SIGNAL_ARRAY+CHFSL_SIG_ARGS>(SP),R3
00F1 799      ; Get signal array size
00F1 800      INCL R3 ; Make loop count correct
52 10 AE43 D6 00F5 801      MOVAL PACK_L_SIGNAL_ARRAY(SP)[R3],R2 ; R2 points beyond exception PSL
F8 A2 FC A0 D0 00F7 802      MOVL -4(R0),-8(R2) ; Unmodified return PC is exception PC
51 50 D0 0101 803      MOVL R0,R1 ; Start writing signal array at return PC
0104 804
0104 805      BRB 40$ ; Join common exit at top of loop
0106 806
0106 807      .DISABLE LOCAL_BLOCK
```



```

0106 809      .SUBTITLE      VAX$REFLECT_TRAP - Reflect Arithmetic Traps
0106 810      :+
0106 811      : Functional Description:
0106 812      :
0106 813      : This routine receives control from the various instruction specific
0106 814      : emulator routines in order to reflect arithmetic traps back to the
0106 815      : caller. There are three arithmetic traps that can occur.
0106 816      :
0106 817      : Decimal overflow
0106 818      :
0106 819      : This can occur in most of the decimal instructions and EDITPC
0106 820      : when there is not enough room in the destination string to
0106 821      : store all of the nonzero digits in the source.
0106 822      :
0106 823      : Integer overflow
0106 824      :
0106 825      : CVTPL can incur this exception when the input decimal string
0106 826      : converts into a longword that cannot fit into 31 bits.
0106 827      :
0106 828      : Divide by zero
0106 829      :
0106 830      : DIVP generates this exception when the divisor is zero.
0106 831      :
0106 832      : Input Parameters:
0106 833      :
0106 834      : 00(SP) - Arithmetic Trap Code (from $SRMDEF)
0106 835      : 04(SP) - PSL on exit from VAX$xxxxxx routine
0106 836      : 08(SP) - Return PC from VAX$xxxxxx routine
0106 837      :
0106 838      : The return PC will determine whether an exception frame (signal array)
0106 839      : already exists or must be built. Briefly, if 08(SP) is equal to the
0106 840      : address called VAX$EXIT_EMULATOR, then the emulator was entered as
0106 841      : a result of execution of a reserved instruction and a signal array
0106 842      : already exists. If 08(SP) is anything else, then it is treated as the
0106 843      : address of an instruction following a JSB into the emulator.
0106 844      :
0106 845      : Implicit Input:
0106 846      :
0106 847      : If 08(SP) is VAX$EXIT_EMULATOR, then the rest of the stack that is
0106 848      : relevant looks like this.
0106 849      :
0106 850      : 12(SP) - Opcode of reserved instruction
0106 851      : 16(SP) - PC of reserved instruction
0106 852      : 20(SP)
0106 853      : .
0106 854      :
0106 855      : 48(SP)
0106 856      : 52(SP) - PC of next instruction
0106 857      : 56(SP) - PSL of original exception
0106 858      :
0106 859      : Output Parameters:
0106 860      :
0106 861      : 00(SP) - Count of longwords in signal array (always equal to 3)
0106 862      : 04(SP) - Signal name (modified form of trap code)
0106 863      : 08(SP) - PC of next instruction
0106 864      : 12(SP) - PSL on exit from VAX$xxxxxx routine
0106 865      :

```



```
0106 866 : Implicit Output:
0106 867 :
0106 868 : This routine passes control to EXE$REFLECT, which will eventually
0106 869 : reflect the arithmetic trap back to the user.
0106 870 :-
0106 871 :
0106 872 : Insure that architectural status codes for arithmetic traps are consistent
0106 873 : with the VMS status codes that they are mapped onto.
0106 874 :
0106 875 ASSUME SS$_INTOVF EQ <SS$_ARTRES + <8 * SRMSK_INT_OVF_T>>
0106 876 ASSUME SS$_FLTDIV EQ <SS$_ARTRES + <8 * SRMSK_FLT_DIV_T>>
0106 877 ASSUME SS$_DECOVF EQ <SS$_ARTRES + <8 * SRMSK_DEC_OVF_T>>
0106 878 :
0106 879 VAX$REFLECT_TRAP::
51 08 AE 50 7D 0106 880 MOVQ R0,-(SP) ; Get some scratch registers
51 00000474 8F C5 0109 881 MULL3 #8,8(SP),R1 ; Turn trap code into status
50 0000 CF 9E 010E 882 ADDL2 #SS$_ARTRES,R1 ; ... by adding in base status code
10 AE 50 D1 0115 883 MOVAB VAX$EXIT_EMULATOR,R0 ; This allows address comparisons
18 13 011A 884 CMPL R0,16(SP) ; Compare with return PC
011E 885 BEQL 10$ ; Some signal array already exists
0120 886 :
0120 887 :+
0120 888 : This code path is taken if the emulator was entered in any way other than
0120 889 : by executing one of the reserved instructions. We assume that the longword
0120 890 : on the stack represents a return PC, the address of the next instruction
0120 891 : that will execute.
0120 892 :
0120 893 : R0 - Scratch
0120 894 : R1 - Modified Trap Code
0120 895 :
0120 896 : 00(SP) - Saved R0
0120 897 : 04(SP) - Saved R1
0120 898 : 08(SP) - Space for trap code
0120 899 : 12(SP) - PSL on exit from VAX$xxxxxx routine
0120 900 : 16(SP) - Return PC
0120 901 :-
0120 902 :
OC 08 AE 51 DO 0120 903 MOVL R1,8(SP) ; Store exception code
50 0C AE DO 0124 904 MOVL 12(SP),R0 ; Save PSL in R0 to start switch
AE 10 AE DO 0128 905 MOVL 16(SP),12(SP) ; Put PC into proper place
10 AE 50 DO 012D 906 MOVL R0,16(SP) ; Store new exception PSL
50 8E 7D 0131 907 MOVQ (SP)+,R0 ; Restore saved R0 and R1
03 DD 0134 908 PUSHL #3 ; Store size of signal array
14 11 0136 909 BRB VAX$REFLECT_TO_VMS ; Join common exit to VMS
0138 910 :
0138 911 :+
0138 912 : This code path is taken if the emulator was entered as a result of executing
0138 913 : one of the reserved instructions. There is lots of extra space on the stack
0138 914 : to fool around with so the juggling act exhibited by the previous block of
0138 915 : code is not necessary here.
0138 916 :
0138 917 : R0 - Scratch
0138 918 : R1 - Modified Trap Code
0138 919 :
0138 920 : 00(SP) - Saved R0
0138 921 : 04(SP) - Saved R1
0138 922 : 08(SP) - Initial trap code (no longer needed)
```



```
0138 923 : 12(SP) - PSL on exit from VAX$xxxxxx routine
0138 924 : 16(SP) - VAX$EXIT_EMULATOR (no longer needed)
0138 925 : 20(SP)
0138 926 :
0138 927 :
0138 928 : 56(SP)
0138 929 : 60(SP) - PC of next instruction (preserved by this code)
0138 930 : 64(SP) - PSL of original exception (with modified condition codes)
0138 931 :
0138 932 : The condition codes that were generated by the VAX$xxxxxx routine are
0138 933 : stored in the exception PSL.
0138 934 :-
0138 935
40 AE 04 00 0C AE F0 0138 936 10$: INSV 12(SP),#0,#4,<EXCEPTION_PSL+20>(SP)
013F 937 ; Store new condition codes
38 AE 51 D0 013F 938 MOVL R1,<OPERAND_8+20>(SP) ; ... and modified trap code
50 8E 7D 0143 939 MOVQ (SP)+,R0 ; Restore saved registers
5E 30 AE 9E 0146 940 MOVAB <OPERAND_8+12>(SP),SP ; Eliminate unneeded stack space
03 DD 014A 941 PUSHL #3 ; Store signal array size and ...
014C 942 ; drop through to VAX$REFLECT_TO_VMS
```



```
014C 944 .SUBTITLE VAX$REFLECT_TO_VMS - Let VMS Reflect the Exception
014C 945 :+
014C 946 : Functional Description:
014C 947 :
014C 948 : This routine is the common exit path to the VMS exception dispatcher.
014C 949 : It executes in three different sets of circumstances.
014C 950 :
014C 951 : 1. Software detected exceptions
014C 952 :
014C 953 : There are several exceptions that are detected by software.
014C 954 : Exception-specific and context-specific routines build a signal array
014C 955 : on the stack. This code then passes that signal array to VMS.
014C 956 :
014C 957 : 2. Modified hardware exceptions
014C 958 :
014C 959 : Certain forms of access violation are modified to appear as if they
014C 960 : occurred at the site of a reserved instruction rather than within the
014C 961 : emulator. Any extraneous stack storage is removed. The PC within the
014C 962 : access violation signal array is modified. This code then passes the
014C 963 : modified signal array to VMS.
014C 964 :
014C 965 : 3. All unmodified exceptions
014C 966 :
014C 967 : Certain exceptions cause the emulator to receive control, even though
014C 968 : the exception in question will be passed intact to VMS. These include
014C 969 : exceptions caused by a random control transfer into the emulator and
014C 970 : access violations such as stack overflow that would not have occurred
014C 971 : in the first place if the reserved instructions were being executed by
014C 972 : the base machine rather than by a software emulator.
014C 973 :
014C 974 : Input Parameters:
014C 975 :
014C 976 : The signal array is on the stack of the access mode in which the
014C 977 : exception occurred.
014C 978 :
014C 979 : 00(SP) - Number of signal array elements (called N)
014C 980 : 04(SP) - Signal name (integer value)
014C 981 : 08(SP) - First exception-specific parameter (if any)
014C 982 : 12(SP) - Second exception-specific parameter (if any)
014C 983 :
014C 984 :
014C 985 : <8+4*N>(SP) - Exception PC that will be reported
014C 986 : <12+4*N>(SP) - Exception PSL that will be reported
014C 987 :
014C 988 : Implicit Output:
014C 989 :
014C 990 : Control is passed to the VMS exception dispatcher at label EXE$REFLECT.
014C 991 : -
014C 992 :
014C 993 VAX$REFLECT_TO_VMS::
014C 994 JMP G^EXE$REFLECT ; Let VMS handle this exception
0152 995
0152 996 .END
```

00000000'GF 17



VAX\$HANDLER  
Symbol table

M 16

- Condition Handlers for VAX-11 Instruct 16-SEP-1984 01:36:04 VAX/VMS Macro V04-00 Page 23  
5-SEP-1984 00:45:37 [EMULAT.SRC]VAXHANDLR.MAR;1 (11)

...OPCODE	= 00000038			OPS_MOVTC	= 0000002E		
ADDP4_B_DELTA_PC	= 00000003			OPS_MOVTUC	= 0000002F		
ADDP6_B_DELTA_PC	= 00000003			OPS_MULP	= 00000025		
ASHP_B_DELTA_PC	= 00000003			OPS_SCANC	= 0000002A		
CHFSL_MCHARGEST	= 00000008			OPS_SKPC	= 0000003B		
CHFSL_MCH_ARGS	= 00000000			OPS_SPANC	= 0000002B		
CHFSL_MCH_SAVRO	= 0000000C			OPS_SUBP4	= 00000022		
CHFSL_SIGARGLST	= 00000004			OPS_SUBP6	= 00000023		
CHFSL_SIG_ARGS	= 00000000			OPCODE_BASE	= FFFFFFFF8		
CMPC3_B_DELTA_PC	= 00000003			OPCODE_MAX	= 0000003B		
CMPC5_B_DELTA_PC	= 00000003			OPERAND_8	= 00000024		
CMPP3_B_DELTA_PC	= 00000003			PACK_L_SAVED_SP	= 00000010		
CMPP4_B_DELTA_PC	= 00000003			PACK_L_SIGNAL_ARRAY	= 00000010		
CRC_B_DELTA_PC	= 0000000B			PACK_L_SIGNAL_ARRAY_POINTER	= 00000018		
CVTLP_B_DELTA_PC	= 0000000B			PACK_V_ACCVIO	= 00000009		
CVTPL_B_DELTA_PC	= 00000003			PACK_V_FPD	= 00000008		
CVTPS_B_DELTA_PC	= 00000003			PSL\$V_FPD	= 0000001B		
CVTPT_B_DELTA_PC	= 00000003			PSL\$V_TP	= 0000001E		
CVTSP_B_DELTA_PC	= 00000003			SCANC_B_DELTA_PC	= 00000003		
CVTTP_B_DELTA_PC	= 00000003			SKPC_B_DELTA_PC	= 00000003		
DELTA_PC_TABLE_BASE	= 00000000 R 04			SPANC_B_DELTA_PC	= 00000003		
DELTA_PC_TABLE_SIZE	= 00000044			SRMSK_DEC_OVF_T	= 00000006		
DIVP_B_DELTA_PC	= 00000003			SRMSK_FLT_DIV_T	= 00000004		
EDITPC_B_DELTA_PC	= 0000000A			SRMSK_INT_OVF_T	= 00000001		
EXCEPTION_PSL	= 0000002C			SS\$_ARTRES	= 00000474		
EXESREFLECT	***** X 00			SS\$_DECOVF	= 000004A4		
JSB_ABSOLUTE	= 00009F16			SS\$_FLTDIV	= 00000494		
JSB_SIZE	= 00000006			SS\$_INTOVF	= 0000047C		
LOCC_B_DELTA_PC	= 00000003			SS\$_RADRMOD	= 0000044C		
MATCHC_B_DELTA_PC	= 00000003			SS\$_ROPRAND	= 00000454		
MOVP_B_DELTA_PC	= 00000003			SUBP4_B_DELTA_PC	= 00000003		
MOVTC_B_DELTA_PC	= 0000000B			SUBP6_B_DELTA_PC	= 00000003		
MOVTUC_B_DELTA_PC	= 0000000B			UNKNOWN	= 000000F1 R 05		
MULP_B_DELTA_PC	= 00000003			VAX\$AL_DELTA_PC_TABLE	= 00000008 RG 04		
NEW_PC	= 00000028			VAX\$BEGIN	= 00000000 R 02		
NO_SIGNAL_ARRAY	= 000000D3 R 05			VAX\$END	= 00000000 R 03		
OLD_PC	= 00000004			VAX\$EXIT_EMULATOR	***** X 00		
OPS_ADDP4	= 00000020			VAX\$MODIFY_EXCEPTION	= 00000000 RG 05		
OPS_ADDP6	= 00000021			VAX\$RADRMOD	= 00000051 RG 05		
OPS_ASHP	= 000000F8			VAX\$REFLECT_FAULT	= 00000073 RG 05		
OPS_CMPC3	= 00000029			VAX\$REFLECT_TO_VMS	= 0000014C RG 05		
OPS_CMPC5	= 0000002D			VAX\$REFLECT_TRAP	= 00000106 RG 05		
OPS_CMPP3	= 00000035			VAX\$ROPRAND	= 0000005D RG 05		
OPS_CMPP4	= 00000037						
OPS_CRC	= 0000000B						
OPS_CVTLP	= 000000F9						
OPS_CVTPL	= 00000036						
OPS_CVTPS	= 00000008						
OPS_CVTPT	= 00000024						
OPS_CVTSP	= 00000009						
OPS_CVTTP	= 00000026						
OPS_DIVP	= 00000027						
OPS_EDITPC	= 00000038						
OPS_JSB	= 00000016						
OPS_LOCC	= 0000003A						
OPS_MATCHC	= 00000039						
OPS_MOVP	= 00000034						



+-----+  
! Psect synopsis !  
+-----+

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 ( 0.)	00 ( 0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
\$ABSS	00000000 ( 0.)	01 ( 1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
_VAX\$\$\$BEGIN	00000000 ( 0.)	02 ( 2.)	PIC USR CON REL LCL SHR EXE RD NOWRT NOVEC PAGE
_VAX\$ END	00000000 ( 0.)	03 ( 3.)	PIC USR CON REL LCL SHR EXE RD NOWRT NOVEC BYTE
_VAX\$DATA	00000044 ( 68.)	04 ( 4.)	PIC USR CON REL LCL SHR NOEXE RD NOWRT NOVEC LONG
_VAX\$CODE	00000152 ( 338.)	05 ( 5.)	PIC USR CON REL LCL SHR EXE RD NOWRT NOVEC QUAD

+-----+  
! Performance indicators !  
+-----+

Phase	Page faults	CPU Time	Elapsed Time
Initialization	17	00:00:00.07	00:00:00.46
Command processing	76	00:00:00.52	00:00:05.45
Pass 1	326	00:00:12.74	00:00:35.67
Symbol table sort	0	00:00:01.86	00:00:03.81
Pass 2	178	00:00:03.09	00:00:08.09
Symbol table output	12	00:00:00.10	00:00:00.16
Psect synopsis output	2	00:00:00.03	00:00:00.03
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	611	00:00:18.41	00:00:53.69

The working set limit was 1350 pages.  
70076 bytes (137 pages) of virtual memory were used to buffer the intermediate code.  
There were 70 pages of symbol table space allocated to hold 1287 non-local and 8 local symbols.  
996 source lines were read in Pass 1, producing 19 object records in Pass 2.  
47 pages of virtual memory were used to define 45 macros.

+-----+  
! Macro library statistics !  
+-----+

Macro library name	Macros defined
_\$255\$DUA28:[EMULAT.OBJ]VAXMACROS.MLB;1	30
-\$255\$DUA28:[SYSLIB]STARLET.MLB;2	10
TOTALS (all libraries)	40

1432 GETS were required to define 40 macros.

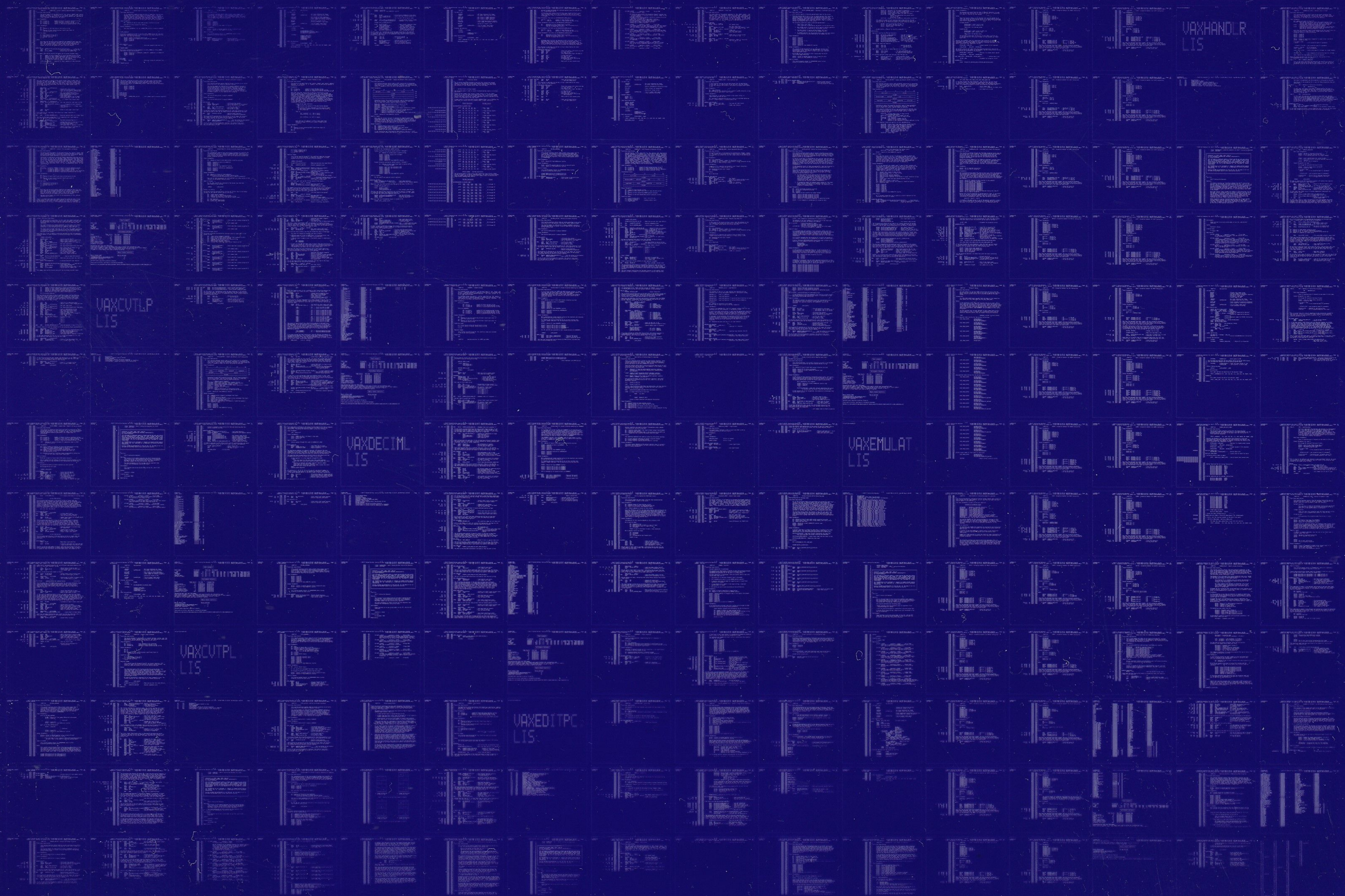
There were no errors, warnings or information messages.

MACRO/LIS=LIS\$:VAXHANDLR/OBJ=OBJ\$:VAXHANDLR MSRC\$:VAXHANDLR/UPDATE=(ENH\$:VAXHANDLR)+LIB\$:VAXMACROS/LIB



0144 AH-BT13A-SE  
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION  
CONFIDENTIAL AND PROPRIETARY





0145 AH-BT13A-SE  
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION  
CONFIDENTIAL AND PROPRIETARY

